|  |  |
| --- | --- |
| 교육 제목 | Machine Learning |
| 교육 일시 | 2021. 10. 20 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish=pd.read\_csv('https://bit.ly/fish\_csv')  fish['Species'].value\_counts()  fish.head()  fish\_input=fish[["Weight", "Length", "Diagonal", "Height", "Width"]].to\_numpy()  fish\_target=fish["Species"].to\_numpy()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  bream\_smelt\_ind=(train\_target=="Bream") | (train\_target=="Smelt")  train\_bream\_smelt=train\_input[bream\_smelt\_ind]  target\_bream\_smelt=train\_target[bream\_smelt\_ind]  print(train\_bream\_smelt.shape)  print(target\_bream\_smelt.shape)  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_bream\_smelt)  train\_scaled=ss.transform(train\_bream\_smelt)  train\_target=target\_bream\_smelt  from sklearn.linear\_model import LogisticRegression  lr=LogisticRegression()  lr.fit(train\_scaled, train\_target)  lr.score(train\_scaled, train\_target)  print(lr.predict(train\_scaled[:5]))  print(train\_target[:5])  print(lr.predict\_proba(train\_scaled[:5]).round(3))  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_input)  train\_scaled=ss.transform(train\_input)  test\_scaled=ss.transform(test\_input)  lr=LogisticRegression()  lr.fit(train\_scaled, train\_target)  print(lr.score(train\_scaled, train\_target))  print(lr.score(test\_scaled, test\_target))  print(lr.predict(test\_scaled[10:20]))  print(test\_target[10:20])  prob=lr.predict\_proba(test\_scaled[:5])  print(prob.round(3))  ----------------------------------------------------------------------------  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish=pd.read\_csv('https://bit.ly/fish\_csv')  fish['Species'].value\_counts()  fish.head()  fish\_input=fish[["Weight", "Length", "Diagonal", "Height", "Width"]].to\_numpy()  fish\_target=fish["Species"].to\_numpy()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_input)  train\_scaled=ss.transform(train\_input)  test\_scaled=ss.transform(test\_input)  from sklearn.linear\_model import SGDClassifier  sg=SGDClassifier(loss="log", max\_iter=1000)  sg.fit(train\_scaled, train\_target)  print(sg.score(train\_scaled, train\_target))  print(sg.score(test\_scaled, test\_target))  sg.partial\_fit(train\_scaled, train\_target)  print(sg.score(train\_scaled, train\_target))  print(sg.score(test\_scaled, test\_target))  ----------------------------------------------------------------------------  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  wine=pd.read\_csv('https://bit.ly/wine-date')  wine.head()  # wine.info()  data=wine[["alcohol", "sugar", "pH"]].to\_numpy()  target=wine["class"].to\_numpy()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  data, target, stratify=target)  print(train\_input.shape)  print(test\_input.shape)  from sklearn.tree import DecisionTreeClassifier  dt=DecisionTreeClassifier()  dt.fit(train\_input, train\_target)  print(dt.score(train\_input, train\_target))  print(dt.score(test\_input, test\_target))  from sklearn.tree import plot\_tree  plt.figure(figsize=(10,7))  plot\_tree(dt, max\_depth=2, feature\_names=['alcohol', 'sugar', 'pH'])  plt.show()  dt=DecisionTreeClassifier(max\_depth=3, random\_state=42)  dt.fit(train\_input, train\_target)  print(dt.score(train\_input, train\_target))  print(dt.score(test\_input, test\_target))  plt.figure(figsize=(20,15))  plot\_tree(dt, filled=True,  feature\_names=['alcohol', 'sugar', 'pH'])  plt.show()  dt.feature\_importances\_ |
| 오후 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  wine=pd.read\_csv('https://bit.ly/wine-date')  wine.head()  data=wine[["alcohol", "sugar", "pH"]].to\_numpy()  target=wine["class"].to\_numpy()  print(data.shape)  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  data, target, stratify=target, random\_state=42)  from sklearn.tree import DecisionTreeClassifier  dt=DecisionTreeClassifier(random\_state=42)  from sklearn.model\_selection import cross\_validate  from sklearn.model\_selection import StratifiedKFold  splitter=StratifiedKFold(n\_splits=10, shuffle=True)#, random\_state=42)  score=cross\_validate(dt, train\_input, train\_target, cv=splitter)  # score=cross\_validate(dt, train\_input, train\_target, cv=10)  score['test\_score'].mean()  print(score)  from sklearn.model\_selection import GridSearchCV  dt=DecisionTreeClassifier()  papam={'max\_depth' : np.arange(4,20, 1), #[2,3,4,5,6],  'min\_impurity\_decrease' : np.arange(0.0001, 0.001, 0.001)}  gs=GridSearchCV(dt, param\_grid=papam, cv=13, n\_jobs=-1)  gs.fit(train\_input, train\_target)  gs.cv\_results\_['mean\_test\_score']  # Cross Validation 에서만 한 것  dt=gs.best\_estimator\_  print(dt.score(train\_input, train\_target))  # 데이터 전체에서 한 것이라 값이 다름  gs.best\_estimator\_  plt.plot(gs.cv\_results\_['mean\_test\_score'])  plt.show()  ---------------------------------------------------------------------------  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  wine=pd.read\_csv('https://bit.ly/wine-date')  wine.head()  data=wine[["alcohol", "sugar", "pH"]].to\_numpy()  target=wine["class"].to\_numpy()  print(data.shape)  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  data, target, stratify=target)  from sklearn.model\_selection import cross\_validate  from sklearn.ensemble import RandomForestClassifier  from sklearn.ensemble import GradientBoostingClassifier  rf=RandomForestClassifier(n\_estimators=100,  criterion="gini",  n\_jobs=-1)  scores=cross\_validate(rf, train\_input, train\_target, cv=10,  return\_train\_score=True)  print(np.mean(scores['test\_score']))  print(np.mean(scores['train\_score']))  rf=RandomForestClassifier(n\_estimators=100,  criterion="gini",  n\_jobs=-1)  rf.fit(train\_input, train\_target)  print(rf.feature\_importances\_)  print(wine.columns)  -------------------------------------------------------------------------  ! pip install wget  ! python -m wget <https://bit.ly/fruits_300> -o fruits\_300.npy  import numpy as np  import matplotlib.pyplot as plt  fruits=np.load('fruits\_300.npy')  fruits[0]  plt.imshow(fruits[10], cmap="gray\_r")  plt.show()  print(fruits.shape)  fruits\_2d=fruits.reshape(-1, 10000)  print(fruits\_2d.shape)  fig, axs=plt.subplots(1,3)  axs[0].imshow(fruits[0], cmap='gray\_r')  axs[1].imshow(fruits[100], cmap='gray\_r')  axs[2].imshow(fruits[200], cmap='gray\_r')  plt.show()  from sklearn.cluster import KMeans  km=KMeans(n\_clusters=3, random\_state=42)  km.fit(fruits\_2d)  print(km.labels\_)  km\_center=km.cluster\_centers\_.reshape(-1, 100, 100)  print(km\_center.shape)  fig, axs=plt.subplots(1,3)  axs[0].imshow(km\_center[0], cmap='gray\_r')  axs[1].imshow(km\_center[1], cmap='gray\_r')  axs[2].imshow(km\_center[2], cmap='gray\_r')  plt.show()  inertia=[]  for k in range(2, 7):  km=KMeans(n\_clusters=k, random\_state=42)  km.fit(fruits\_2d)  inertia.append(km.inertia\_)    plt.plot(inertia)  plt.show()  km.predict(fruits\_2d[100:101]) |